Defense Acquisition Performance: Could Some Agility Help?

Paul E. McMahon
PEM Systems

The problems with the Waterfall Model and the challenges involved with evolutionary acquisition of defense systems have been well-chronicled. This article describes how Agile techniques can be applied to address these challenges under today's defense acquisition regulations. The article employs a previously published hypothetical space system acquisition case study, along with a case study of a legacy modernization project that employed Agile techniques to aid the acquisition process.

In the May 2007 CrossTalk article, "Defense Acquisition Performance Assessment – The Life-Cycle Perspective of Selected Recommendations," Dr. Peter Hantos analyzes the *conceptual integrity* of recent defense acquisition recommendations made in a Defense Acquisition Performance Assessment (DAPA) report. Hantos states that:

> The acquisition life-cycle models of the DoD/National Security Space Acquisition Policy 03-01 policies are inherently Waterfall, and as such, inadequate for the acquisition of large-scale, software-intensive systems even if they are used with the intent of Evolutionary Acquisition. [1]

Much has been written over the past few years about the problems with the Waterfall Model, so the issues raised by Hantos are not surprising. Through the use of a hypothetical space system acquisition case study employing an evolutionary acquisition strategy, the following three key obstacles are identified and discussed in the primary sections of this article:

1. Funding for Risk Mitigation Alternatives.
2. Deferring Non-Key Performance Parameter Requirements.
3. The Inability to Define Measures of Technology Readiness.

Over the past few years, Agile practices have also received attention mostly on small commercial projects in the software arena, although interest has recently been growing in the defense industry and on larger efforts [2]. This article describes how Agile techniques can help with the evolutionary acquisition of defense systems. In particular, this article explains how an Agile approach could be used to effectively tackle each of the three obstacles identified in the referenced article.

1. Funding for Risk Mitigation Alternatives

Traditional risk management approaches identify, categorize, and prioritize risks, along with establishing risk mitigation approaches. If the risk is high enough, alternatives may be pursued through distinct parallel activities (used as a backup in case the current course fails). Often, however, risk mitigation is handled through the normal engineering management activities without the need for added funding for alternatives.

Hantos also states that:

> Having risk mitigation plans in the conventional sense is different from spiral planning. It involves the creation of additional plans to eliminate or gradually reduce the risk by having alternative course(s) of action lined up in case the risk materializes or its likelihood drastically increases. A key element of such risk planning is that funding for alternative actions needs to be provided in addition to the allocated, regular cost of development. [1]

How Do Agile Teams Mitigate Risk?

Successful Agile teams that I have observed deal with risk mitigation differently [2]. Rather than expend valuable resources on alternative approaches that may never get used, resources are focused on breaking the known problem down into manageable chunks and tackling each to systematically gain knowledge that reduces uncertainty. With the new knowledge gained, course corrections are acted upon in a timely fashion.

Key to Agile planning is not setting the plan in stone for the subsequent increment prior to assessing the new knowledge gained from the previous increment. Agile approaches provide risk mitigation by reducing the risk through a series of small course corrections rather than spending extra resources on distinct alternatives that may never be employed. The Agile approach does, however, require an understanding of how to continuously identify the current high-priority work that needs to be done now based on the latest knowledge of the risk, and what work to *defer* given the current available resources [3]. This leads to the second obstacle.

2. Deferring Non-Key Performance Parameter Requirements

In the case study described in [1], difficulties encountered when attempting to defer work on a space system legacy modernization project are analyzed:

> Allowing program managers to defer non-key performance parameter requirements to later upgrades is an attractive proposition from the program manager's view. It provides an effective risk management tool by greatly expanding their decision-making authority and flexibility. In the context of our case study, how could the program manager using this newly acquired freedom reduce the scope of the first acquisition increment? Unfortunately, analysis shows there are not many opportunities after all. [1]

The challenges encountered when planning and executing the modernization of a critical legacy application are not insignificant. For example, Hantos alludes to the challenges related to convincing an end-user to incrementally accept partially modernized functionality when they need all of their current legacy functionality to do their job today. This leads to a question: Does the evolutionary acquisition concept fall apart when applied to modernizing tightly coupled critical legacy systems?

Deferring work is fundamental to Agile risk management practices. But the decision to defer work—and deciding what

work to defer—isn't made by the program manager alone. As the following different case study demonstrates, the key value of work deferment isn't in providing "freedom" to the program manager to "reduce the scope" of any particular increment of the project.

A Different Case Study Using Agile Practices to Modernize a Business-Critical Legacy System

Over the past few years, I was on a team that helped a client modernize a legacy system that is critical to their business. Using Agile practices, we attempted to plan an incremental approach to deploy the modernized functionality.

We started with the user interface. It was a chunk that could be broken off and integrated with the remaining legacy applications supporting incremental deployment. Some argued against this approach because the user interface was not viewed as the highest technical risk. With Agile practices, we attack high risks first—but technical risks are not necessarily the highest risks. In fact, we soon discovered that end-user buy-in was a higher risk, especially when modernizing a legacy system that is critical to an organization's daily functions and has been in place for 40 years. We also discovered that by providing a modern user interface first—and by not only demonstrating this capability early, but also training end-users early and incrementally while listening intently to their feedback—we were able to build key relationships and a key support group that became crucial to keeping the project on firm ground and moving forward.

On the legacy modernization case study, similar to the space system case study article, the majority of the legacy application subsystems were too tightly coupled to be deployed in separate increments. But we also knew that due to the criticality of the full system (to the business) that it needed to be fully tested and accepted by the end-users prior to deployment. To meet this challenge, it was decided to continue with an evolutionary life cycle but deploy it incrementally to a lab environment where the new system could be tested and demonstrated, and where end-users could both provide feedback to developers and be given training.

Each increment of work culminated in a week-long lab session where previously agreed-to user scenarios were demonstrated and end-users were given hands-on operational training. It was during the hands-on lab time when end-users provided their most valuable feedback to the development team. These collaborative sessions proved invaluable as end-users became familiar with features that would simplify their current job, while also finding *acceptable alternative approaches* to non-key requirements that had been *deferred* to later increments. It is important to note that by providing this *incremental training* before all requirements had been implemented that many deferred *non-key requirements* were mutually agreed to be *no longer needed.*

As the end-user's knowledge and confidence with the new system grew through each lab session, they became the most powerful voice and driving factor in the eventual full acceptance and deployment of the modernized system. Formal training was deployed to all end-users prior to the system go-live date, and an aggressive maintenance support program was put in place that included an around-the-clock help desk and on-site, immediate technical support.

> *"As the end-user's knowledge and confidence with the new system grew ... they became the most powerful voice and driving factor in the eventual full acceptance and deployment of the modernized system."*

The legacy modernization case study was similar to the hypothetical space system study in that there were not many opportunities to deploy *incrementally to-the-field*, but this fact did not reduce the importance of employing an incremental evolutionary life cycle. This approach was critical to our ultimate success. Also critical was the strategy of integrating end-users into the engineering cycle early where their voice, along with the voice of the development team, became a driving force in decisions on what work to defer and what work to pull forward. These decisions were coordinated through the program manager.

Also critical to the project's success was management's commitment of the time required by end-users to fully participate in the multiple lab sessions. The time it takes to learn a new system cannot be underestimated nor its importance from the end-user buy-in perspective. This may well have been the most important risk-mitigation strategy employed. Without the strong commitment of the end-users, who truly felt they were a part of the project development team, this effort could not have succeeded.

It is worth noting that while the legacy modernization case study described may not technically fit the definition of evolutionary acquisition, it demonstrates the criticality of an *evolutionary development strategy* (that includes integrated end-user collaboration) to the success of tightly coupled legacy modernization projects.

A Key Value of Requirements Deferral

It is also important to note that a key value in deferring work when using Agile practices is to provide the opportunity to the development team to maximize value to the end-users, rather than providing the program manager with freedom to reduce incremental scope.

This key value is often misunderstood. Some view the primary value of requirements deferral as a method to allow program managers to defer facing difficult cost and schedule challenges. Actually, its greatest value is the opposite. Requirements deferral, when used appropriately, can help address critical risks sooner (e.g., the risk of end-user buy-in) while at the same time increasing the opportunity to meet user needs more cost-effectively through alternative approaches (as shown in the legacy modernization case study). Nevertheless, applying requirements deferral appropriately can be tricky, as the following examples demonstrate.

An example of an appropriate use of requirements deferral is to free up the needed human resources from a less important task, pulling a more valuable task forward in the schedule to help in demonstrating a key system feature sooner to an end-user. When this is done appropriately, earned value (EV) and schedule commitments are maintained since we are trading the value of one task for another task of equal or greater importance. The overall project risk is also reduced.

With respect to the space system study, Hantos states:

> There are other considerations that would make the deferral of requirements difficult. For example, complex graphics and elaborate

> display designs are important in any ground system. As a requirements-pacing strategy, one might consider releasing the first version of the ground software with simplified user interfaces. This is an effective engineering approach, but it may backfire with end-users of the system. In similar situations, satellite operators forced to work with intermediate systems having limited capabilities created resentment and blocked buy-in when the final system became available. [1]

If the lab approach (as discussed in the legacy modernization case study) was used in the space system case study, this resentment might have been avoided since the operators would not be forced to use the limited capabilities in an operational environment. By training the operators early in the lab environment, there is a good possibility they could find alternative approaches to non-key deferred requirements—as was the case in our legacy modernization project. The feedback from the operators could also lead to decisions to pull critical display requirements forward, while deferring less critical user interface functions and thereby aiding overall end-user buy-in.

But what if you are already behind schedule and don't have the resources to pull any work forward in the schedule? In this situation, it can be appropriate to defer requirements to meet a critical milestone. Nevertheless, it is important to understand when using this technique that there is a difference between meeting a milestone and being on schedule from an EV perspective.

For example, it is inappropriate to defer requirements and then take EV for work that was never completed, thereby creating the false impression of being on schedule from an EV perspective. When less work is accomplished than was planned, you are behind schedule from an EV perspective. Unfortunately, too often today, program managers inappropriately utilize requirements deferral in this manner to mask the real project status.

Decisions related to pushing work out and pulling other work in should be made by key knowledgeable technical team members together with management and the customer. These decisions should be made at the start of each increment to ensure the best decision is made with the best available information and to ensure that all stakeholders understand and are on board with the rationale for a requirement deferral decision.

3. The Inability to Define Measures of Technology Readiness

According to Hantos, the DAPA report's findings state that:

> ... there are no clearly definable measures of technology readiness, and the inability to define and measure technology readiness during Technology Readiness Assessments is the reason that immature technology is incorporated into plans prior to milestone B. [1]

Hantos responds by stating: "On the contrary, numerous sources are available to help with technology readiness assessments" [1]. Hantos also provides a number of sources currently available to support this position.

"By training the operators early in the lab environment, there is a good possibility they could find alternative approaches to non-key deferred requirements—as was the case in our legacy modernization project."

While I don't disagree with the existence of these sources, I believe the underlying intent of the DAPA report was to raise awareness to the lack of effectiveness of these sources in achieving their ultimate goal of providing an accurate assessment of technology readiness.

How Do Successful Agile Teams Measure Technology Readiness?

Agile practices employ a simple approach to technology readiness. The following criteria give a sense for this approach [3, 4]:

- The work associated with the technology can be estimated by those who are going to do the work.
- Those who are going to do the work have established at least one implementation approach.
- The planned work can be partitioned into reasonably small chunks (one to three months).

This approach is not meant to imply that the best estimates come only from practitioners. To the contrary, especially on large efforts; accurate estimates require both a bottom-up and top-down approach. Many Agile development estimation techniques out-of-the-box don't scale well to large projects because they focus primarily on a bottom-up estimation approach, which is insufficient for large complex efforts [4]. However, a hybrid approach composed of *Agile bottom-up* together with a more traditional top-down approach as a cross-check has been proven to work well for large-scale efforts [2, 3].

Some believe that on complex efforts with long development cycles that it is impractical to perform detailed (daily/weekly) cost and schedule estimates. I agree that it doesn't make sense to formally update the project master plan and schedule daily or even weekly on large complex efforts. It has been my experience, however, that the most successful large-scale efforts manage the work daily at the grassroots level and continuously update their estimates using informal daily or weekly meetings and team task lists [2]. With the successful projects, these meetings are short and directly involve the people who are actually doing the real work—with management primarily in a listening and supporting role.

It is important to recognize that my recommendations are not meant to imply that we should wait until a technology is mature, or within months of being mature, before incorporating it. This would lead us back to a Waterfall approach where we need to know everything before beginning. This is why I recommend a hybrid Agile-traditional approach for large complex efforts.

Unfortunately, I am finding that many are missing the fact that Agile approaches are not in conflict with high-risk technologies. In fact, they are best suited to address high risks because, when applied appropriately, Agile approaches drive the true cost and schedule to the surface sooner.

Some may also believe this approach is too simplistic for the complex work involved with space systems. But does the intent of measuring technology readiness change based on complexity? If its intent relates to establishing a confidence factor in hitting cost and schedule targets when implementing the technology, then I

believe the right balance of Agile practices is the best choice.

The technology readiness criteria used on Agile projects go beyond just the need for granular work packages. Those who are going to do the work must have an approach and must have completed adequate *analysis* to commit to the *collaboratively agreed to* cost and schedule targets to complete that work.

To accurately estimate effort, regardless of complexity, the problem needs to be broken down into manageable chunks that can be estimated by those who are actually going to do the work—not their managers. Agile practices utilize the knowledge that people are the most important factor in hitting cost and schedule commitments, and these practices recognize that different people perform in different ways.

By requiring that at least one implementation approach be known, I am saying that sufficient *analysis* needs to have been completed during the initial planning stage to ensure that the problem has been thought through enough to see a potential reasonable solution.

On large complex efforts, however, bottom-up and top-down estimates rarely get to the same point. This is not necessarily bad. Having a *challenge* schedule can be good to help stretch the team's productivity. But when the difference between the published schedule and the developer's real work gets too wide, this strategy can backfire, leading to the wrong decisions by the wrong people at the wrong time.

Final Thoughts

We should not wait to have all the answers before utilizing new technologies. On the other hand, experiences on both Agile and non-Agile efforts of the past indicate that the lack of adequate analysis may lie at the root of immature technologies being prematurely incorporated into project plans [2]. It is an appropriate balance we seek.

Contractors have more options today under the current defense acquisition regulations than many realize. Agile methods can be successfully used in DoD acquisitions today; but to succeed, it is important to have management buy-in and an understanding of the implications of using the right balance of Agile techniques on both the government and contractor side.

Agile approaches will not solve all the current acquisition challenges faced today, nor will they make a complex problem simple. However, the greatest benefit of Agile approaches may not be in solving small, well-understood problems, but in helping solve those that are least understood. This is because they help us avoid making decisions too early (e.g., leaning toward the Waterfall approach) and too late (e.g., deferring the wrong work for the wrong reasons). Agile practices—balanced appropriately with proven traditional practices—can help us understand sooner how big a problem is, and can help us leverage today's available options and human resources more effectively, regardless of the problem's complexity.◆

References

1. Hantos, Peter. "Defense Acquisition Performance Assessment – The Life-Cycle Perspective of Selected Recommendations." CROSSTALK May 2007.
2. McMahon, Paul E. "Lessons Learned Using Agile Methods on Large Defense Contracts." CROSSTALK May 2006.
3. McMahon, Paul E. "Are Management Basics Affected When Using Agile Methods?" CROSSTALK Nov. 2006.
4. Cohn, Mike. Agile Estimating and Planning. Pearson Education, Inc., 2006.

About the Author

Paul E. McMahon, principal of PEM Systems, helps large and small organizations as they move toward increased agility and process maturity. He has taught software engineering, conducted workshops on engineering process and management, is a frequent speaker at industry conferences, and is a certified ScrumMaster. McMahon has published articles on Agile development and lessons using the CMMI® framework as well as the book, "Virtual Project Management: Software Solutions for Today and the Future." He has more than 25 years of engineering and management experience and 10 years of independent consulting experience.

PEM Systems
118 Matthews ST
Binghamton, NY 13905
Phone: (607) 798-7740
E-mail: pemcmahon@acm.org

® CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.